Apple IIe
#4: RDY Line

Revised by: Glenn A. Baxter November 1988
Written by: Peter Baum July 1984

This Technical Note describes an input signal to the 6502 microprocessor called the RDY line.

Using the RDY Line on the Apple IIe and Apple][+

Though the 6502 was one of the first commercially successful microprocessors sold, the designers had foresight to include some very useful functions. Because many early peripherals products were very slow devices, a microprocessor could not read from the device directly. To connect these slow devices onto the Apple peripheral bus so the 6502 can read data from them requires either buffering the device or slowing down the processor. Though most people would try to buffer the device, sometimes it is not feasible. When buffering is not possible, a peripheral device can pull the RDY line to slow down the processor long enough to read a byte. This technique can be used by slow devices to communicate with the 6502.

The RDY line allows a peripheral card to halt the microprocessor during read operations (opcode, operand, or data fetches—reads) with the output address lines reflecting the current address being fetched. If a peripheral device cannot get data on the bus fast enough to meet the setup time of the 6502, then the peripheral card can pull the RDY line low and tell the 6502 to wait. This **cannot** be done during a 6502 write cycle because the 6502 does not wait during writes.

For the 6502 to read a valid data byte from a peripheral card, the card has about 800 ns from the time the addresses are valid to put the data on the bus. The data must be setup on the bus within approximately 400 ns from the time that the I/O STROBE, I/O SELECT, or DEVICE SELECT signal on the peripheral slot goes true. If a device pulls the RDY line low for one clock cycle, the device will have approximately 1.4 μs, instead of the 400 ns, to put out valid data. The RDY line can be pulled low for more than one cycle—in fact, there is no limit. A device that takes 100 μs to send data can just hold the RDY line low for 100 cycles. Hence, this technique will allow any slower device to get on the bus and send data to the 6502.

This is a bit different than DMA on the Apple IIs. DMA actually prevents the CPU from receiving a clock signal, whereas the RDY line is actually a function of the processor. In Apple II DMA, the 6502 CPU will die after approximately 15 clocks because it depends on the clock to refresh its internal registers. (The 6502 is dynamic, whereas the 65C02 is static, and therefore

not affected by the absence of clock information). In the case of the RDY line, the CPU is internally told to just not complete its bus cycle until RDY is de-asserted. This is a similar concept to DTACK on the Motorola 68000 series CPUs.

The RDY line is typically pulled low during PH1, but the specification sheets for the 6502 show that it can be pulled anytime before the last 200 ns of PH2. The PH2 line is not used by the Apple II and is an unused output from the 6502. It is basically the same as the PH0 line with a little delay. Before I explain when to use (or not use, in some cases) the RDY line, let us first look at some timing diagrams of the Apple system.

Figure 1 shows the relationship between the 6502 and Apple IIe and Apple][+. The timing specifications have been adjusted to reflect the signals as they are seen from the peripheral slots. For example the 6502 (1 MHz) specification guarantees that the address bus will be valid within 225 ns from PH2 out. But the peripheral slots do not see these address lines directly. Instead, the address lines go through a buffer and then out to the peripheral slots. This routing adds a maximum delay of 13 ns in the Apple][and 18 ns in the Apple IIe. The timing diagrams will show, in the case of an Apple][, that the address bus will be valid to the peripheral slots within 238 ns (225+13) of the PH2 falling edge.

The major differences in timing between the Apple][+ and the Apple IIe are due to the processor. The Apple][uses a 1 MHz 6502, while the Apple IIe uses a 6502A, which is a 2 MHz part. This does not mean that the system clock in the Apple IIe runs any faster, only that the 6502A is capable of running faster. This difference results in better timing margins. For example, the address and data buses are set up faster in the Apple IIe by the 6502A than the 6502 sets them up in the Apple][. (This was done because the custom chips in the Apple IIe are slower than the discrete logic in the Apple][, and the 6502A was needed to compensate).

A peripheral card which uses the RDY line can only be used under certain circumstances. Because pulling the RDY line low halts the processor, any program with a software timing loop may not work properly. These programs assume that each instruction will take a fixed amount of time, which is not true when the processor stops in the middle of an instruction. An Apple II Disk is an example of a peripheral which requires timing loops and may not run properly if the RDY line is used.

	Apple][1 MHz 6502		Apple IIe 2 MHz 6502A	
Symbol	**Minimum**	**Maximum**	**Minimum**	**Maximum**
T02- *	15	50+20 (LS08)	15	50+5 (S02)
T02+ *	30	80+15 (LS08)	30	80+5 (S02)
Tads		225+13 (8T97)		140+18 (LS244)
Trwh	30		30	
Tdevsel—		96 (3 x LS138)		65 (LS154+LS138)
Tiosel—		64 (2 x LS138)		38 (LS138)
Tiostb—		32 (LS138)		15 (LS10)
Tdevsel+		18 (LS138)		30 (LS154)
Tiosel+		36 (2 x LS138)		18 (LS138)
Tiostb+		18 (LS138)		15 (LS10)
Tdsu	100+17 (8T28)**		50+12 (LS245)	

Thr	10	10
Trs ***	200	200

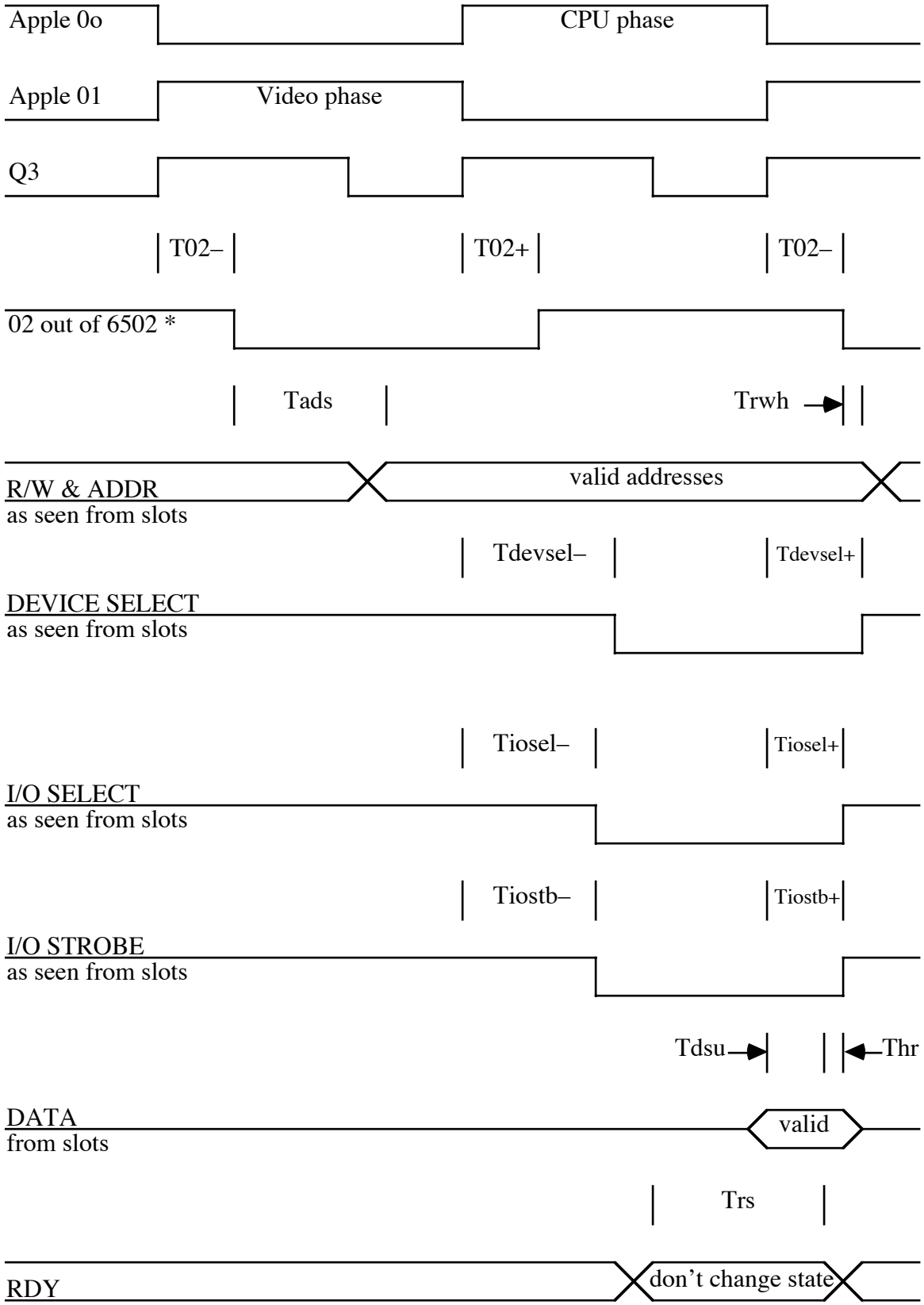
(All times are given in nanoseconds (ns).)

* load = 100 pf.

** The RFI versions of the Apple][+, revisions A through D motherboards, use an 8304 instead an 8T28.

*** The RDY line must never change states within T_{rs} to end of 02.

Table 1–Timing Specifications for Figure 1



* – 02 is an output signal from the 6502 which is not used by the Apple. It is a delayed 0o.

Figure 1–Timing Signals As Seen From the Peripheral Slots

Table 1 lists three different type of numbers. If a number is by itself, then it is just the corresponding 6502 or 6502A specification. If a number is followed by parenthesis, then it represents the delay, produced by TTL gates, between the 6502 and the peripheral slots. The characters in the parenthesis denote the part number(s) of the part(s) which generated the delay. These parts are typically 74' series TTL except for the 8T28 and 8T97. If there are two numbers in a column with a plus sign (+) then the first number signifies the 6502 specification and the second the TTL delay, with the corresponding part number. Most of the TTL delay times are from the Texas Instrument data books. The 6502 specifications are from the Synertek 6502 data sheet and from Synertek application note AN2 - SY6500.

When the RDY Line Can be Changed and When It Cannot

As can be seen from these figures, the RDY line should not be gated with the PH0 trailing edge since this happens around the same time as the falling edge of PH2. This would violate the T_{RS} specification and probably force the 6502 to perform erratically. Gating the RDY line with the trailing edge of Q3 during PH0 might work, but this could leave as little as 25 ns for the signal to be valid. In other words, Q3 must enable the RDY line low within 25 ns of Q3 changing states. If this output cannot be guaranteed stable, then the RDY line might violate the T_{RS} specification.

The safest time to pull the RDY line is using the PH0 rising edge, but this edge occurs before I/O SELECT, I/O STROBE, or DEVICE SELECT are enabled. Therefore, this scheme will not work if any of these three enables is used by the peripheral card. For example, many peripheral cards use memory mapped I/O to transfer data with the cards registers designed to reside in the DEVICE SELECT memory space. Location $C0n0 (where n = 8 + slot number of peripheral card) might hold the status of the card, and location $C0n1 might be used to read a device such as a disk or an A/D converter. The card uses the DEVICE SELECT signal, pin 41 on the slot, and the four low-order address lines to determine if the 6502 wants to read the status register or read from the A/D converter. Typically, the status register can put its data on the bus within 200 ns, easily meeting the setup requirements of the 6502. But the A/D converter might take at least 100 μs before it can respond with data. The RDY line must be pulled low to allow time for the A/D converter to set up the data bus. Notice that the peripheral card does not know that it should pull the RDY line low until after the DEVICE SELECT signal has gone low. This signal does not go low until after PH0 goes high, so the PH0 rising edge cannot be used to enable the RDY line for this peripheral card.

There are a few ways around this problem. One solution would be to decode the $C0n0 address on the peripheral card and not use DEVICE SELECT. This solution also requires either putting user-selectable switches on the card for setting the slot number, or making the card slot dependent. Another solution is to pull the RDY line low using one of the first three edges, trailing or leading, of the 7 M clock. These edges occur at 70, 140, and 210 ns into PH0 and are trailing, leading, then trailing edges, respectively. The best solution is to use the DEVICE SELECT signal to enable the RDY line. Figure 2 should help.

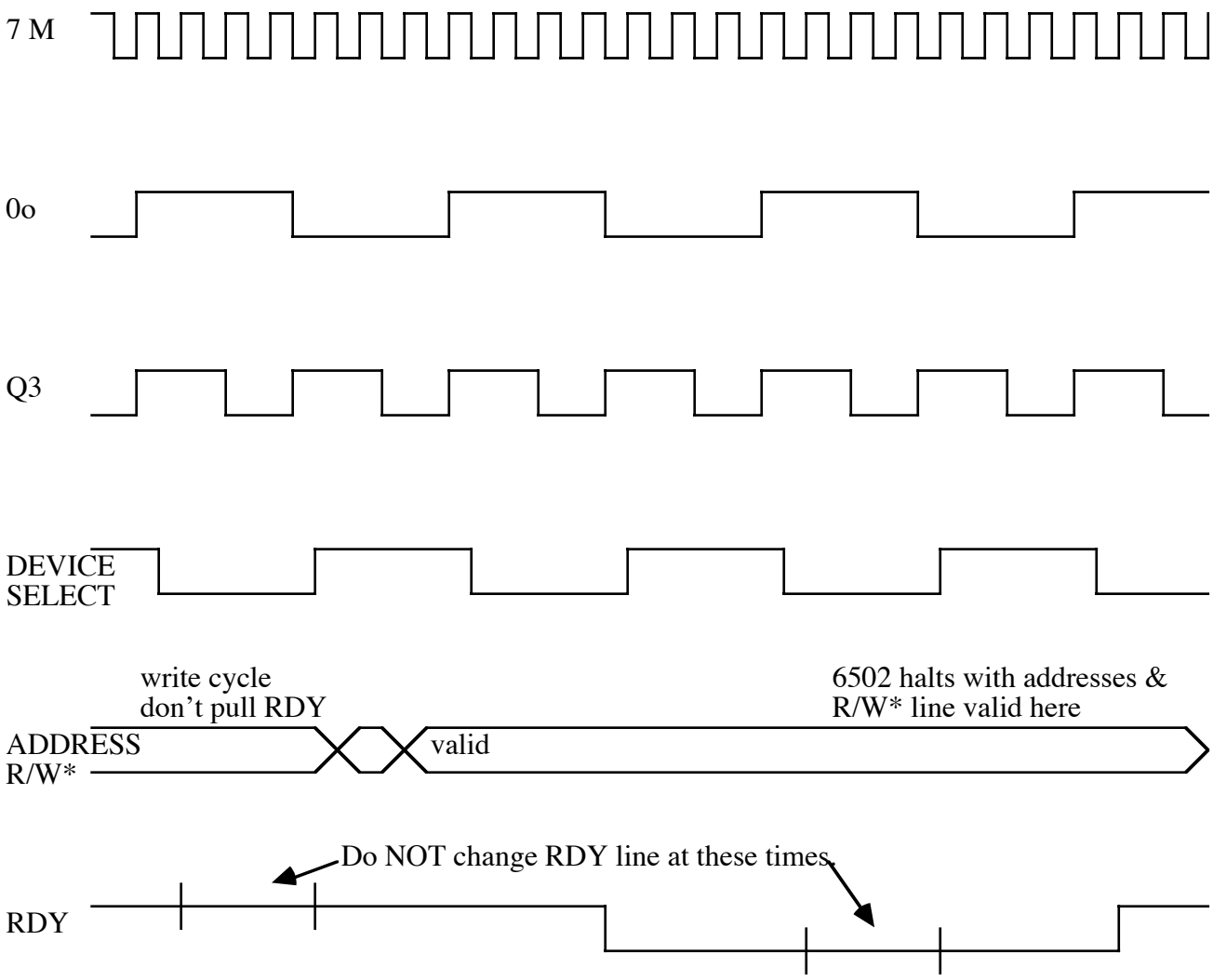


Figure 2–Timing Diagram

Do Not Pull RDY During Write Cycles

Because there is no acknowledge response from the 6502, the peripheral card must do some of its own housekeeping and check if a write cycle is taking place. On write cycles, the 6502 will not halt, but continue running until the next read cycle. After a slow peripheral pulls the RDY line and before it tries to get on the bus, it must make sure the 6502 is not in the middle of a write cycle. Otherwise there will be a bus crash, as both the peripheral card and 6502 try to drive the bus. One simple way to prevent this bus crash from occurring is to make sure the peripheral card does not pull the RDY line low during a write cycle. You can guarantee this will not happen by checking the R/W line when PH0 goes high or DEVICE SELECT goes low. The R/W line is guaranteed to be stable by this time.

Releasing the RDY Line

When the RDY line is released, the 6502 will continue the cycle that was originally halted and allow the 6502 to read the data bus. Data will be read on the next trailing edge of PH2 by the 6502, as long as RDY does not change within T_{RS} of the end of PH2. When the peripheral device has set the data bus up with the correct data, it can release the RDY line to complete the read cycle. Releasing the RDY line has exactly the same constraints as pulling the line; do not change RDY within 200 ns of the end of PH2.

The RDY line can be released before data has been set up, if the data will be valid within specification. This means that RDY can be released in the middle of PH1 if the data bus will be valid 117 ns before PH2 trailing edge, for the Apple][(62 ns for the Apple IIe).

Slow Writes

Since the 6502 cannot be halted during write cycles, if a device requires longer than one cycle to receive data then the data must be buffered. Here is an example of how to accomplish this:

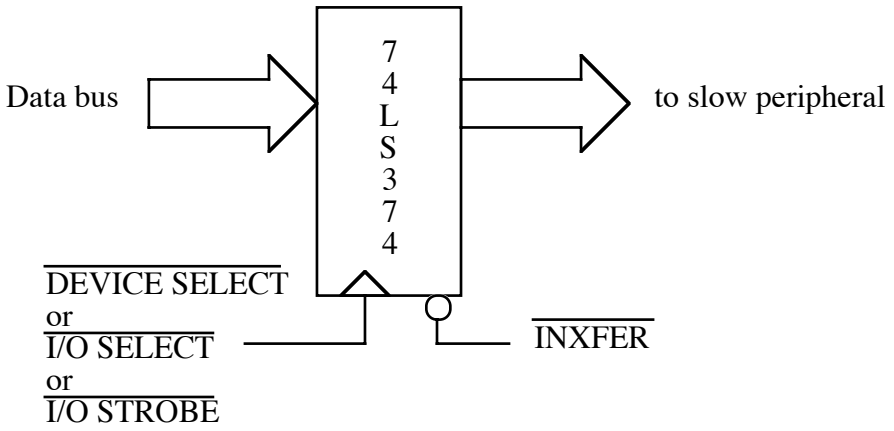


Figure 3–Buffering Data

Note: It is very easy to overrun the slow peripheral using this scheme, since it only buffers one byte at a time. Do not send data twice to the buffer within the maximum allowed time between slow peripheral reads.

Further Reference

- *Apple IIe Technical Reference Manual*